Pareto-based Grouping Discrete Harmony Search Algorithm for Multi-objective Flexible Job Shop Scheduling[1]

K.Z. Gao[a,b], P.N. Suganthan[a], Q.K. Pan[b], T.J. Chua[c], T.X. Cai[c], C.S. Chong[c]

[a] *School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore*
[b] *Computer school, Liaocheng University, Liaocheng, 252000, PR China*
[c] *Singapore Institute of Manufacturing Technology, Nanyang Drive 638075, Singapore*

Abstract: This paper proposes a Pareto-based grouping discrete harmony search algorithm (PGDHS) to solve the multi-objective flexible job shop scheduling problem (FJSP). Two objectives, namely the maximum completion time (makespan) and the mean of earliness and tardiness, are considered simultaneously. Firstly, two novel heuristics and several existing heuristics are employed to initialize the harmony memory. Secondly, multiple harmony generation strategies are proposed to improve the performance of harmony search algorithm. The operation sequence in a new harmony is produced based on the encoding method and the characteristics of FJSP. Thirdly, two local search methods based on critical path and due date are embedded to enhance the exploitation capability. Finally, extensive computational experiments are carried out using well-known benchmark instances. Three widely used performance measures, number of non-dominated solutions, diversification metric and quality metric, are employed to test the performance of PGDHS algorithm. Computational results and comparisons show the efficiency and effectiveness of the proposed PGDHS algorithm for solving multi-objective flexible job-shop scheduling problem.

Keywords: Flexible job shop scheduling, Harmony search algorithm, Multi-objective optimization, Local search

1. Introduction

The flexible job-shop scheduling problem (FJSP) is a branch of the job-shop scheduling problem (JSP) with extensive practical applications [10, 15]. The FJSP is an NP-hard problem [10, 16]. The FJSP includes two sub-problems: (a) machine assignment that is to select a machine from a set of candidate machines for operations; and (b) operation sequence that is scheduling the operations on all machines to obtain a feasible solution. Thus, the FJSP is harder than the classical JSP. Generally, the objectives considered in FJSP include the maximal completion time (makespan), the total work load of machines, the maximal machine workload, the mean/total earliness time and tardiness time compared to the due date of jobs. Earliness and tardiness objectives are due date related criteria. The FJSP problem has wide application in manufacturing engineering, such as turning operations [42, 43] and milling operations [44, 45].

Brucker and Schile [2] developed a polynomial graphical algorithm for FJSP with two jobs known to be the first work for FJSP. Due to the complexity of FJSP, meta-heuristic algorithms have become the preferred solution techniques to produce good schedules in a reasonable time. Multi-objective FJSP has been studied by many researchers in recent years. Xia and Wu [39] proposed a hierarchical solution approach by using a particle swam optimization (PSO) algorithm to assign operations on machines and a simulated annealing (SA) algorithm to schedule operations on each machine. Gao [6] employed a genetic algorithm which hybrids with a variable neighborhood descent algorithm for the multi-objective FJSP. Xing et al [40] proposed a simulation model for solving multi-objective FJSP. This model is a general framework for the problem enabling the applications of different algorithms. Xing et al [41] also employed a local search algorithm for the multi-objective total FJSP (T-FJSP) and partial FJSP (P-FJSP). The results show that the algorithms are efficient for FJSP problem, especially for the large scale FJSP problems. Zhang et al [46] researched a hybrid PSO and tabu search (TS) algorithm to solve FJSP with multi-objective. Li [20] proposed a hybrid TS algorithm for multi-objective FJSP. In this

[1] Corresponding author. Tel.: +65 83845266.

E-mail address: kzgao@ntu.edu.sg (K.Z.Gao), epnsugan@ntu.edu.sg (P.N. Suganthan), qkpan@gmail.com (Q.K. Pan), tjchua@simtech.a-star.edu.sg (T.J. Chua).

research, a variable neighborhood search algorithm and a local search method are integrated. Additionally, Li [21] also employed a Pareto-domination based artificial bee colony (PDABC) algorithm for the multi-objective FJSP. An external Pareto archive set is designed in this paper to record the non-dominated solutions. And a fast non-dominated set update process is introduced in the algorithm. Wang [34] proposed an enhanced Pareto-based artificial bee colony (EPABC) algorithm for the multiple objectives FJSP. A local search strategy based on critical path is embedded in the search framework to enhance the local intensification capability. The results and comparisons show that the performance of EPABC is better than that of PDABC.

To solve the multi-objective FJSP, researchers primarily considered some combinations of the makespan, the maximal machine workload and total workload as objectives. A few researchers also considered due date related criteria: mean/total earliness time and tardiness time. Scrich et al [32] considered the total tardiness in the FJSP and solved using a tabu search based algorithm. Wu and Weng [38] proposed a multi-agent method to optimize the earliness and tardiness objectives in FJSP. Gholami [13] minimized the makespan and mean tardiness objectives in a dynamic and flexible job shop. Vilcot [33] optimized the makespan and maximum lateness objectives in a flexible job shop environment from printing and boarding industry. Fattahi [5] optimized the makespan and total weighted tardiness objectives.

Harmony search (HS) is a relatively recent meta-heuristic method developed by Geem et al [11] for solving optimization problems. It imitates the music improvisation process of musicians. A harmony in music is regarded analogous to a solution vector in the corresponding optimization problem, while the musician's improvisations are regarded analogous to local and global search processes in optimization algorithms. The HS algorithm generates a new candidate solution by making use of all existing solutions. However, compared to other meta-heuristics, the HS algorithm imposes fewer mathematical requirements. Numerical comparisons also show that evolution in HS is faster than that of the GA [24, 26].

In recent years, many researchers have developed improved HS algorithms and applied HS variants to solve scheduling problems [18]. Omran and Mahdavi [26] developed a global-best harmony search algorithm. Das [3] analyzed and improved the exploration power of HS algorithm. Geem applied HS algorithm for scheduling multiple water distribution systems [12]. Wang [35] and Pan [27] proposed a hybrid harmony search algorithm and a local-best harmony search algorithm with dynamic sub-harmony memories for solving blocking flow shop scheduling problem and lot-streaming flow shop scheduling problem. Gao proposed a grouping harmony search algorithm [8] and a discrete harmony search (DHS) algorithm [7] for solving no-wait flow shop scheduling problem. Following the successful applications of HS algorithm, we proposed a Pareto-based DHS algorithm for flexible job shop scheduling problem [9]. This work developed a Pareto-based grouping discrete harmony search (PGDHS) algorithm to solve FJSP with makespan, the mean of earliness and tardiness criteria. In this paper, several existing heuristics were employed for initializing the harmony memory. A new heuristic was proposed for initializing harmony memory. A grouping strategy was used for improving the convergence. A new local search approach was proposed for minimization of earliness and tardiness. In addition, more benchmark instances were tested with detailed discussions and comparisons. Consequently, the performance of the proposed PGDHS algorithm is substantially improved beyond the conference paper [9]. The rest of this paper is organized as follows: In Section 2, we briefly describe the problem formulation of FJSP. The basic HS algorithm is introduced in Section 3. Section 4 presents the PGDHS algorithm in detail. Section 5 presents the experimental results and compares with other algorithms in existing literature to demonstrate the superior performance of the proposed PGDHS algorithm. Finally, we conclude the paper in Section 6.

2. Multi-objective flexible job shop problem

2.1 Multi-objective optimization

A multi-objective optimization problem can be described as follows:

$$\text{Min } f(x) = (f_1(x), f_2(x), \cdots, f_n(x)), \quad x \in \Omega, \quad f(x) \in R^n \tag{1}$$

where x is the decision vector in space Ω and $f(x)$ is the objective vector.

Pareto domination states that solution A dominates solution B if and only if $\forall i \in \{1,2,\cdots,n\}, f_i(A) \le f_i(B)$ and $\exists i \in \{1,2,\cdots,n\}, f_i(A) < f_i(B)$. Solution A is an optimal in the Pareto sense if there is not any solution B which dominates A. Pareto optimal set is the collection of all Pareto optimal solutions and the corresponding image in the objective space is the Pareto front. In this paper, an archive set (AS) is used to record the non-dominated solutions during the iterations. During the search process, if one new solution dominates one or more solutions in AS, the new solution will replace the dominated solutions in AS.

For multi-objective optimization, the developed algorithm should obtain more non-dominated solutions with good proximity and diversity [31, 36] with respect to the true Pareto front. In other words, it requires the algorithm to obtain more non-dominated solutions, on or closer to the optimal Pareto front, and distributed evenly over the whole Pareto front.

2.2 Formulation of multi-objective FJSP

In a flexible job shop, each job has a sequence of operations. An operation can be executed by one machine only out of a set of candidate machines. Each operation of a job can be processed by only on one machine at a time, while each machine can process only one operation at a time. The FJSP can be divided into two categories: total FJSP (T-FJSP) and partial FJSP (P-FJSP) [17]. In T-FJSP, each operation can be processed on all machines in the workshop. In P-FJSP, at least one operation cannot be processed on all machines. Hence, it is clear that the P-FJSP is harder to solve than the T-FJSP when all other factors are the same. Thus, the T-FJSP is a special case of P-FJSP.

In FJSP, the makespan criterion has been previously considered by other researchers. The due date related criterion is important for manufacturing enterprises with just-in-time (JIT) production environments. The motivations for implementing JIT production are to reduce inventories, improve response times, cash flows and customer satisfaction. In JIT production, earliness means excessive inventories while tardiness means late product deliveries to customers. Thus, in this study, we consider minimizing the makespan and mean of earliness and tardiness of FJSP. These objectives are in conflict with each other to some extent and have been used in some literatures [10, 30]. Our mathematical model is comparable to the model used in [21]. The following notations and assumptions are used in the formulation of multi-objective FJSP.

1) Let $J = \{J_i\}, 1 \le i \le n$, with index i, be a set of n jobs to be scheduled. q_i denotes total number of operations of job J_i, d_i is the due date of J_i.
2) Let $M = \{M_k\}, 1 \le k \le m$, with index k, be a set of m machines.
3) Each job J_i has a predetermined sequence of operations. *Let* $O_{i,h}$ be operation h of J_i.
4) Each operation $O_{i,h}$ can be processed without interruption on one of the candidate machines $M(O_{i,h})$. Let $P_{i,h,k}$ be the processing time of $O_{i,h}$ on machine M_k.
5) Decision variables

$$x_{i,h,k} = \begin{cases} 1, \text{if machine } k \text{ is selected for the operation } O_{i,h} \\ 0, \text{otherwise} \end{cases}$$

$c_{i,h}$ denotes the completion time of the operation $O_{i,h}$

c_i denotes the completion time of the job J_i

6) The two objectives to minimize are:
Makespan, denoted by C_M, is the maximal of completion time of machines.

$$\text{Min } C_M = \max_{1 \le i \le n} \{c_i\} \tag{2}$$

Mean of earliness and tardiness, denoted by E/T, is the earliness or tardiness of one job compared to the due date of the job d_i.

$$\text{Min } E/T = \frac{\sum_{i=1}^{n} |c_i - d_i|}{n} \tag{3}$$

3. Harmony search algorithm

The HS algorithm, proposed by Geem [11], was inspired by the natural musical performance process whereby a musician searches for a better state of harmony. In HS, each solution is called a 'harmony' which is an *n* dimensional vector. The basic single objective HS algorithm is summarized as follows:

Step 1: Initialization

The harmony vector or a solution is represented as an *n*-dimensional real-valued vector $X = \{x(1), x(2), ..., x(n)\}$, where each element $x(k)$ denotes a decision variable of the optimization problem. This step sets parameters and fills the harmony memory (HM) with a population of initial harmonies (solutions). Parameters include the harmony memory size (HMS), harmony memory consideration rate ($HMCR$), pitch adjustment rate (PAR) and distance bandwidth (BW). The HM is initialized by randomly generated solution vectors.

Let $X_i = \{x_i(1), x_i(2), .., x_i(n)\}$ represent the i^{th} harmony vector in the HM, which is generated as follows:

$$x_i(j) = LB(j) + (UB(j) - LB(j)) \times rand() \tag{4}$$

for $j = 1,2,..,n$ and $i = 1,2,\cdots HMS$

where $rand()$ is a uniform random number in the range of [0,1], $LB(j)$ and $UB(j)$ are the lower and upper bounds of the decision variable $x(j)$, respectively.

Step 2: Improvising a new harmony

The process of generating a new harmony vector X_{new} is called improvisation. X_{new} is produced by applying three rules: a memory consideration, a pitch adjustment and a random selection. First, if a uniform random number $rand()$ in the range of [0, 1] is less than $HMCR$, the decision variable $x_{new}(j)$ is generated by the memory consideration. Otherwise, $x_{new}(j)$ is obtained by a random selection. Secondly, each decision variable $x_{new}(j)$ will undergo a pitch adjustment with a probability of PAR if it is updated by the memory consideration. The memory consideration, pitch adjustment and random selection are given in Eqs (5), (6) and (7), respectively.

$$\begin{aligned} &if\ (rand1() < HMCR) \\ &\quad x_{new}(j) = x_a(j) \end{aligned} \tag{5}$$

$$\begin{aligned} &if\ (rand2() < PAR) \\ &\quad x_{new}(j) = x_{new}(j) \pm rand() \times BW \end{aligned} \tag{6}$$

$$\begin{aligned} &else \\ &x_{new}(j) = LB(j) + rand() \times (UB(j) - LB(j)) \end{aligned} \tag{7}$$

where $a \in (1,2,..,HMS)$, $x_a(j)$ is randomly selected from HM.

Step 3: Updating the HM

The HM will be updated if X_{new} is better than the worst harmony vector X_w in the HM. In this case, X_{new} will replace X_w and become a member of the HM.

Step 4: Stopping condition

If a termination criterion is met, return the best harmony vector X_B in the HM. If not, go to Step 2.

4. Pareto-based discrete harmony search algorithm for FJSP

4.1 Coding and decoding

A solution consists of two vectors corresponding to the machine assignment and operation scheduling sub-problems of the FJSP. After coding, a harmony is composed of two parts:

- Machine assignment vector (called MA).
- Operation sequence vector (called OS).

Fig. 1 (a) shows a machine assignment (MA) vector while Fig.1 (b) shows the corresponding operation sequence (OS). In MA, each element represents the machine selected for the corresponding operation. In OS, the same elements represent the different operations of the same job [6]. For example, the first "3" in Fig. 1 (a) means that machine 3 is selected for operation $O_{1,1}$. The second "2" in Fig. 1 (b) represents the second operation of job2 while the third "4" is the third operation of job4.

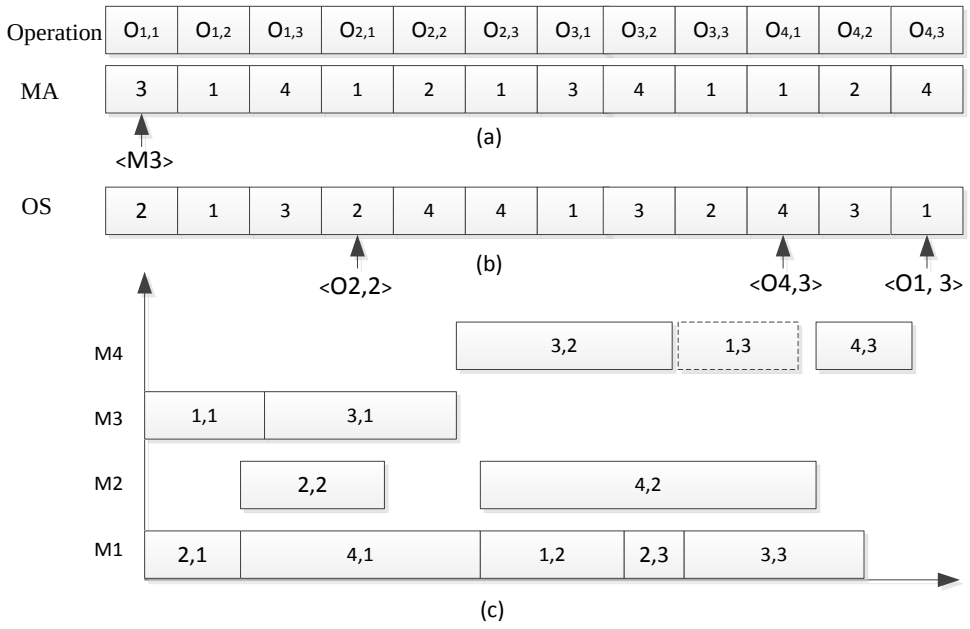

Fig.1 Illustration of MA and OS

Procedure: Find the earliest idle time interval for operation *O*i,j

Step1: Set I=1.

Step2: Count the interval, S, between operations I and operation I+1 on current machine.

Step3: If S is larger or equal the processing time of operation O_{ij}, go to Step5, else go to Step4.

Step4: I=I+1, if I equals to the number of operations the machine has processed, go to Step6; else go to Step2.

Step5: Insert operation O_{ij} between operations I and I+1 and output.

Step6: Insert O_{ij} after the finial operation which has been scheduled and output.

Fig.2 The steps of finding the earliest idle time

Pinedo [29] grouped schedules into three classes: non-delay schedule, active schedule and semi-schedule. It has been verified in this literature that an active schedule contains the optimal schedule. In an active schedule, there is no operation to be processed earlier except putting off another operation's start time or changing the order of operations. Hence, we decode the MAOS solution to an active schedule to reduce the search space. The criterion for obtaining the active schedule is to find the earliest idle time interval for each operation on the processing machine [47]. Fig.2 shows how to find the earliest idle time interval. For example, Fig.1(c) is the Gantt chart decoded from the coding shown in Fig.1 (a)

and Fig.1 (b). It can be seen from Fig.1 (b) that the operation $O_{4,3}$ is decoded before operation $O_{1,3}$. In Fig.1 (a), operation $O_{1,3}$ is processed on M4, the same machine with operation $O_{4,3}$. If the earliest idle time is not considered, the operation $O_{1,3}$ will be assigned after operation $O_{4,3}$. It can be seen from Fig.1 (c) that the end time of $O_{3,2}$ is larger than the end time of $O_{1,2}$ and the interval between operations $O_{3,2}$ and $O_{4,3}$ is larger than the processing time of operation $O_{1,3}$. Hence, operation $O_{1,3}$ can be inserted between operations $O_{3,2}$ and $O_{4,3}$. In this way, the release time of M4 will be the completion time of operation $O_{4,3}$. Inserting operation $O_{1,3}$ on machine M4 does not increase the release time of machine M4.

4.2 Initializing harmony memory

In PGDHS algorithm, the quality of initial HM influences the convergence to a satisfactory solution. Therefore, it is critical to generate a better quality initial HM. The initialization process includes machine assignment and operation scheduling phases. The adoption of a mix of following rules is used to produce the machine assignment and operation scheduling for FJSP problems. For example, the percentages of each initializing rule are equal.

4.2.1 Machine assignment component

Except Random rule, three heuristic rules are employed for initializing machine assignment:

- Global minimum-processing time rule [28]

  This rule considers both the processing time and the workload of machines and start from the operation with the global minimum processing. The processing time is added to the machine workload. For other operations, the machine with minimum processing time (workload) are fixed and assigned. The machine's workload update is also performed. This rule considers the global workload among all machines and can potentially find better makespan. The disadvantage of this approach is the lack of diversity.

- Two-step greedy rule [33]

  The operations are sorted in ascending order based on the number of selectable machines with ascending order of processing times to break ties when there is equal number of selectable machines. The machines are sorted in their workload non-decreasing order. The operation is taken from the operations list and the first machine which belongs to the machine list is assigned to the operation. The workload of this machine is updated and the machine sorting is also updated. The process iterates until all operations have been assigned to machines. This two-step greedy rule is proposed to minimize of the maximum tardiness.

- Proposed minimum-completion time rule

  We develop this rule based on the operations' minimum processing time rule [28]. The proposed rule is as follows:

  Step1: For each operation $O_{i,j}$, select machine M1 with the minimum processing time and machine M2 with the earliest feasible time from selectable machine set.

  Step2: Calculate the completion time T1 and T2 of operation $O_{i,j}$ on machines M1 and M2.

  Step3: If T1<T2, M1 is selected for processing operation $O_{i,j}$; otherwise, M2 is selected for processing operation $O_{i,j}$

  This rule considers the completion of each operation and conducive to optimizing the makespan.

4.2.2 Operation scheduling component

Once the machine assignment is fixed, the operations on each machine should be sequenced. Except random rule, the operation scheduling component is initialized by the following three methods [21]:

- Most work remaining rule

This method firstly orders the jobs in a descending order based on the remaining processing time. The job with the most remaining processing time will be selected first and put into the operation sequence. The iteration is repeated until all operations of all jobs are sequenced.

- Most operations remaining rule
  This rule selects jobs based on the number of remaining operations. The job with the most remaining operations unprocessed is selected first.

- Shortest processing time rule
  In this approach, the operation with the shortest processing time is selected from the next executable operations.

4.2.3 Early end time rule

The initialization methods in Sections 4.2.1 and 4.2.2 do machine assignment component first and perform the operation scheduling after machine assignment. In this section, we propose an initialization rule in the reverse process. The process is as follows:

Step1: The operation sequence is obtained by randomly shuffling the order of all operations of all jobs.

Step2: For each operation $O_{i,j}$ in operation sequence, calculate the end time on all selectable machines.

Step3: The machine with the minimum end time is selected for processing operation $O_{i,j}$.

4.3 Dynamic grouping optimization

The small-sized HM works better than a large one for the HS algorithm [23] and dynamic subpopulation can effectively balance the fast convergence and large diversity [27]. Hence, the PGDSH algorithm employs multiple small-sized dynamic sub-HMs. More specifically, the whole HM of the PGDHS algorithm is grouped into small-sized sub-HMs randomly in first iteration. Then, each sub-HM uses its own members to search for better solutions in the search space. After this iteration, the sub-HMs form a whole HM which is again regrouped into small-sized sub-HMs randomly. And the sub-HMs restart their search independently. This process, shown in Figure 3, is continued until a termination criterion is met.

4.4 Improvising new harmony

In PGDHS algorithm, each harmony includes two parts, machine assignment and operation sequence. In MA part, each element means a machine for processing corresponding operation. In MA part, the elements at the same position are for the same operation while the elements at the same position may mean the operation of different jobs in OS part. Therefore, a new harmony for the machine assignment part and operation sequence part should be improvised differently. The process of improvising a new MA solution is shown in Fig. 4.

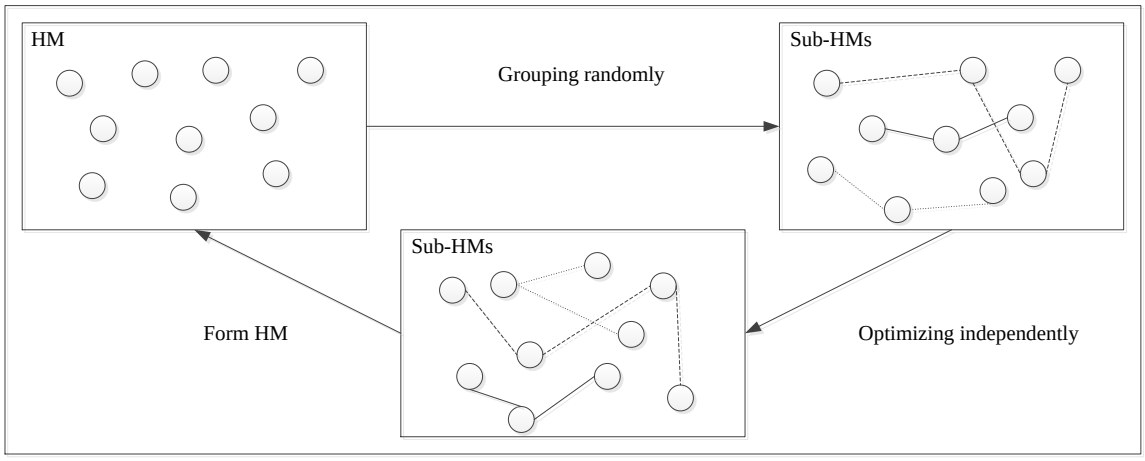

Fig.3 Dynamic grouping process

Procedure: Improving the new MA part for new harmony

Step1: Generate two random numbers *R1* and *R2*.

Step2: If *R1*<*HMCR* and *R2*<*PAR*, go to Step3; else, go to Step6;

Step3: Select one machine *k* for current operation. If *k* is the last machine in the set of candidate machine, go to Step4; else, go to Step5.

Step4: Select the first machine *k'* in the set of candidate machines for current operation.

Step5: Select the next machine *k''* of machine *k* in the set of candidate machines for current operation.

Step6: Randomly select one machine *k'''* from the set of candidate machines for current operation.

Fig.4 Steps for improvising the MA part of a new harmony

Based on the characteristic of FJSP and the coding strategies, we use crossover operators for the OS part of a new harmony. There are several crossover operations proposed during the past decades, such as partial-mapped crossover, order crossover, cycle crossover and so on [6]. In this study, we employ a new crossover operator using order crossover to obtain two new harmonies from two current harmonies. The crossover operator is implemented for the OS part as illustrated in Fig. 5 and summarized below:

Step1: Generate a random number *R* from 1 to the number of jobs;

Step2: Copy the values from the OS part of Harmony1 to the corresponding positions in New Harmony1 where the values are less than or equal to *R*.

Step3: Copy the values from the OS part of Harmony2 to the corresponding positions in New Harmony2 where the values are larger than *R*.

Step4: From the OS part of Harmony2, copy the values which do not appear in New Harmony1 to the vacant positions in New Harmony1 from left to right according to the order of the sequence in Harmony2.

Step5: From the OS part of Harmony1, copy the values which do not appear in New Harmony2 to the vacant positions in New Harmony2 from left to right according to the order of the sequence in Harmony1.

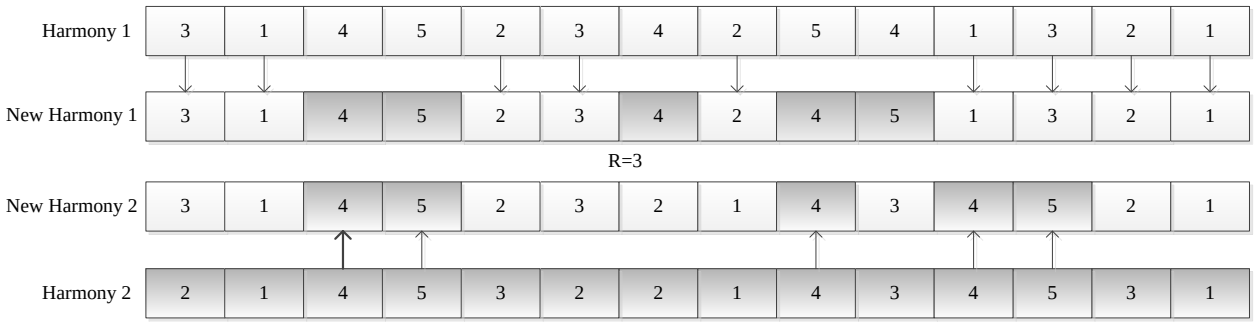

Fig.5 An example of the crossover operation for OS part of a new harmony

In this paper, we proposed an improvisation to generate multiple new harmonies. For each sub-HM, the number of new harmonies is the same as the number of harmonies in this sub-HM. For the OS part, there will be four operation sequences after applying crossover operation in Fig. 5. Meanwhile, we also generate four MA solutions correspondingly. We select two harmonies from the four harmonies based on domination. Two harmonies will be selected randomly if the four harmonies are non-dominated.

4.5 Local search

In this paper, we proposed two local search algorithms for FJSP, one for makespan criterion based on critical path and another for E/T criterion based on due date of jobs.

4.5.1 Local search algorithm for makespan criterion

The critical path concept has been employed in job shop scheduling problem to improve the convergence speed [48]. In this study, we proposed a local search algorithm based on critical path for the makespan criterion of FJSP. However, we also consider the neighborhood from the different selectable machines of public critical operations. In the local search method, if the operation has two or more machines available for selection, we will assign a machine different from the current one. The first operation and the last operation in the public critical block will swap with the adjacent public critical operations. The process of local search algorithm for makespan criterion is shown in Fig. 6. For example, the operations $O_{2,1}$, $O_{4,1}$, $O_{4,2}$ and $O_{4,3}$ in Fig.1(c) are critical operations. The local search algorithm will exchange the positions of $O_{2,1}$, and $O_{4,1}$. If $O_{4,2}$ and $O_{4,3}$ have more than one selectable machines, two machines will be selected to replace current machines for $O_{4,2}$ and $O_{4,3}$.

4.5.2 Local search algorithm for E/T criterion

For earliness-tardiness criterion, the jobs are divided into two groups in the proposed local search algorithm. Group one includes the jobs which are completed before the due date while group two has the jobs which are completed after the due date. For all operations on all machines, the operations which belong to the job in group one move to right adjacent position while the operations which belong to the job in group two move to left adjacent position. More specifically, the operations on the same machine are considered as pair from the first two operations. If the jobs including one pair of operations are included in different groups, the position of the pair of operations on machine will be interchanged. For example, shown in Fig.1(c), there are five operations on machine M1. Suppose the completion time of job 3 is after the due date and the completion time of job 2 is before the due date. The operations $O_{2,3}$ and $O_{3,3}$ will swap their positions.

4.6 Computational procedure

Due to the complexity of flexible job shop scheduling problem, the proposed PGDHS algorithm employs multiple initialization methods, novel improvisation of new harmonies, two local search algorithms for solving the FJSP effectively. The diversity of populations and the balance of global exploration and local exploitation are all considered. The proposed PGDHS algorithm is illustrated in Figure 7. First, harmony memory and archived set are initialized. Then, harmony memory is divided into multiple Sub-HMs. Each Sub-HM executes improvising a new harmony and local search independently. After that, Sub-HMs and AS are updated. If the maximum iteration is not reached, the harmony memory is grouped again. The grouping operator and local search balance the global exploration and local exploitation. The grouping operator and improvising a new harmony stress the diversity of population during the search process. Hence, the PGDHS is expected to yield good performance.

Procedure: local search algorithm

Step1: If critical block *pb* includes one operation, go to Step2; else, go to Step3.

Step2: If the operation has more than one candidate machine, select the machine with least release time to replace the current one.

Step3: Set I=1.

Step4: If the operation I critical block is the first operation in *pb*, swap it with the next operation; if the operation I is the last operation in *pb*, swap it with the ahead operation; else, go to Step5.

Step5: If the operation can be processed by more than one machine, randomly select one machine different to the current one.

Step6: Evaluate all solutions and record the best one.

Step7: Compare all solutions by different public critical block and select the best one.

Fig.6 The steps of local search algorithm

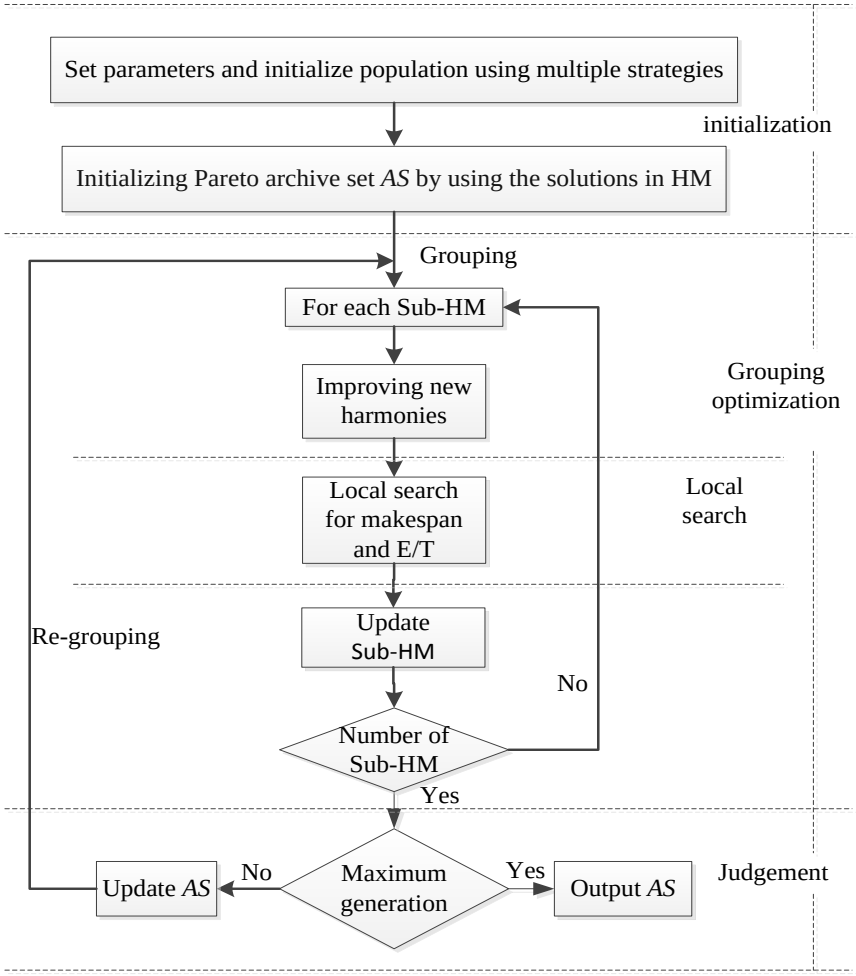

Fig.7 The proposed PGDHS algorithm

5. Computational results and comparisons

5.1 Experimental setup

To evaluate the performance of the proposed PGDHS algorithm, extensive experimental evaluation and comparisons with other methods were conducted and are presented using well-known FJSP benchmark sets. Two sets of problem instances are considered:

- The first data set are five Kacem instances [16, 17].
- The second data set, called BRdata, is a set of 10 problems proposed by Brandimare [1].

The Kacem benchmark set is composed of 5 instances with the size ranging from 4 jobs, five machines and 11 operations to 15 jobs, 10 machines and 60 operations. The Brandimare benchmark set includes 10 problems with the size ranging from 10 jobs, 6 machines and 55 operations to 20 jobs, 16 machines and 240 operations. In order to introduce due date in the data sets, the method inspired by Gholami [13] is employed. The due date of job J_i is defined by

$$\mathrm{D}_i = \left(1 + \frac{T \times n}{m}\right) \times \sum_{j=1}^{n_i} p_{i,j} \tag{8}$$

where T is a parameter that has been fixed as $T = 0.3$, n is the number of jobs, m is the number of machines, $p_{i,j}$ is the average processing time of operation $O_{i,j}$ on all selected machines. Pareto-based multi-objective optimization algorithms are committed to find an approximate of non-dominated solutions. Hence, the performance metrics are different from the

single objective methods. The following widely used performance measures are used in this paper [30] with larger values representing better performances:

- Number of non-dominated solutions (NNdS)
  This performance measure counts the total number of non-dominated solutions generated by the compared algorithms.
- Diversification metric (DM)
  DM determines the diversity of non-dominated solutions by each compared algorithm. Inverted generational distance (IGD) is used by many researchers for assessing the DM performance [4, 14, 19, 49]. Let P^* be a set of uniformly distributed points in the objective space along the Pareto front (PF). Let P be the set of non-dominated solutions by compared algorithms. The inverted generational distance from P^* to P is defined as

$$IGD(P*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P^*|} \tag{9}$$

  where $d(v, P)$ is the minimum Euclidean distance in the objective v and the points in P. If $|P^*|$ is large enough to represent the PF well, $IGD(P*, P)$ could measure both the diversity and convergence of P in a sense. To have a low value of $IGD(P*, P)$, P must be very close to the PF and should not miss any part of the whole PF. In this paper, we set P^* as the set of Pareto solutions in PF and P is the set of non-dominated solutions obtained by each algorithm.
- Quality metric (QM)
  In this metric, a combined overall Pareto front is constructed from all non-dominated solutions obtained by all compared algorithms. Then, the number of Pareto solutions contributed by each comparing algorithm to the overall pareto front is counted and the percentage of the Pareto solutions belonging to each algorithm is also calculated. The algorithm with higher value is better.

Algorithms were coded in C++.net and run on Intel 2.8GHz PC with 1 GB memory. The parameters HMS, HMCR and PAR affect the performance of HS algorithm. Omran and Mahdavi [26] suggested that it was generally better to use a large value of HMCR (i.e., >=0.9). A large PAR value enhances the local exploitation ability of the algorithm while a small PAR value enlarges the search area and diversity of the HS algorithm. In this paper, the parameters are fixed as follows: HMS=1000, HMCR=0.95 and PAR=0.5. Each instance is run for the 20 independent replications with $10 \times n \times m$ generation. In this paper, grouping operator is employed for improving algorithm convergence. We set five levels for the size of Sub-HM and discuss the difference using the problem MK09 instance in Brandimare benchmark set. Instance MK09 was run 20 replications on each level of Sub-HM size. Table 1 shows the average results of performance measures NNS, DM and QM. It is clear that the algorithm has the best results when the size of Sub-HM equals 20. Because HMS is 1000, the number of Sub-HM is 50 in this paper.

Table 1 Level trend of Sub-HM size

Performance Measures	size of Sub-HM				
	10	20	30	50	100
Ave_NNdS	4.65	4.8	4.7	4.6	4.55
Ave_DM	40.15	40.24	39.41	38.04	37.86
Ave_QM	0.93	0.96	0.94	0.92	0.91

5.2 Results and comparisons

To test the performance of our algorithm, the proposed PGDHS algorithm is compared with several recently published algorithms, such as AL+CGA presented by Kacem et al. [17], Multiagent method (Multi-Agent) proposed by Wu and Weng [38], the multi-objective genetic algorithm (MOGA) designed by Wang et al. [37], the particle swarm optimization

and local search algorithm (MoPSO+LS) developed by Moslehi and Mahnam [25], the hybrid shuffled frog-leaping algorithm (HFSLA) proposed by Li et al. [22] and the enhanced Pareto-based artificial bee colony algorithm (EPABC) designed by Wang [34]. The two objectives are considered simultaneously, i.e. minimization of makespan (denoted by C_M) and the mean of earliness and tardiness (denoted by E/T).

5.2.1 Test on five Kacem instances

Table 2 shows the results of five Kacem instances. All Pareto solutions in Table 2 are shown in boldface. It can be easily seen that: 1) Except AL+CGA, all algorithms can obtain the three Pareto solutions, (11, 7), (12, 6) and (16, 2), for instance 4×5. 2) For instance 8×8, MOPSO+LS, HSFLA and EPABC identify two Pareto solutions, (14, 10) and (15, 8) while MOGA obtains three, (14, 10), (15, 8) and (20. 5). But, PGDHS obtains one more Pareto solution, (24, 3). 3) For instance 10×7, PGDHS obtains three Pareto solutions, (11, 13), (12, 9) and (16, 4). MOGA, MOPSO+LS, HSFLA and EPABC obtain just two. 4) AL+CGA obtain three of seven Pareto solutions for instance 10×10 while MOGA, MOPSO+LS, HSFLA and EPABC obtain five. PGDHS discovers all seven Pareto solutions. 5) For instance 15×10, PGDHS obtains four Pareto solutions, (11, 18), (12, 15), (18, 8), (20, 6), more than other compared algorithms. Table 2 also shows the central processing unit (CPU) time of PGDHS algorithm. It also shows that proposed algorithm is efficient since the CPU time cost is low.

Table 3 shows the number of non-dominated solutions and quality metric of all compared algorithms for five Kacem instances. The second column shows the number of Pareto solutions in PF. For each compared algorithm, the total number of non-dominated solutions, the number of non-dominated solutions which belong to the overall combined PF and the average quality metric values for five instances are also calculated. For example, AL+CGA can find two non-dominated solutions for instance 5, but both of them are not in the overall combined PF. For the same instance, MOGA obtained three non-dominated solutions and all of them are in PF while PGDHS algorithm can find four non-dominated solutions and all of them are in the overall combined PF. It can be seen that the proposed PGDHS algorithm has 21 non-dominated solutions and all these non-dominated solutions are in the overall combined PF. The quality metric values obtained by PGDHS are all 100% for five instances. It means that PGDHS can obtain all Pareto solutions for five instances. For above mentioned performance measures, PGDHS has better results than the compared five algorithms.

In addition, the diversification metric values of each compared algorithms are computed for five instances. The minimum, mean and standard deviation of IGD-metric values are shown in Table 4. It can be seen from Table 4 that five algorithms can obtain the best minimum values for five instances except AL+CGA. Among six compared algorithms, only PGDHS found the best mean and standard deviation values for five instances. Hence, the diversification metric of PGDHS algorithm is better than the five compared algorithms.

Hence, the PGDHS is better than compared five algorithms for five Kacem instances. As an example, Figure 8 shows the Gantt chart of Pareto solution (24, 3) of Kacem instance 8×8 in which the due dates of all jobs are 22, 31, 22, 23, 33, 28, 26 and 31.

Table 2 Comparisons of PGDHS with five existing algorithms based on Makespan and E/T objectives.

Instance	AL+CGA		MOGA		MOPSO+LS		HSFLA		EPABC		PGDHS		CPU (s)
(n×m)	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	
4×5	16	6	**11**	**7**	**11**	**7**	**11**	**7**	**11**	**7**	**11**	**7**	0.327
	18	4	**12**	**6**	**12**	**6**	**12**	**6**	**12**	**6**	**12**	**6**	
			16	**2**	**16**	**2**	**16**	**2**	**16**	**2**	**16**	**2**	
8×8	15	11	**14**	**10**	**14**	**10**	**14**	**10**	**14**	**10**	**14**	**10**	2.126
	24	8	**15**	**8**	**15**	**8**	**15**	**8**	**15**	**8**	**15**	**8**	
			20	**5**							**20**	**5**	
											24	**3**	
10×7	12	13	**11**	**13**	**11**	**13**	**11**	**13**	**11**	**13**	**11**	**13**	2.509
	13	9	**12**	**9**	**12**	**9**	**12**	**9**	**12**	**9**	**12**	**9**	
			17	6	17	6	17	6	17	6	**16**	**4**	
10×10	7	13	**7**	**12**	**7**	**12**	**7**	**12**	**7**	**12**	**7**	**12**	3.49
	8	**11**	**8**	**11**	**8**	**11**	**8**	**11**	**8**	**11**	**8**	**11**	

	10	**7**	**10**	**7**	**10**	**7**	**10**	**7**	**10**	**7**	**10**	**7**	
	11	**6**	**11**	**6**	**11**	**6**	**11**	**6**	**11**	**6**	**11**	**6**	
			12	**5**	**12**	**5**	**12**	**5**	**12**	**5**	**12**	**5**	
											15	**4**	
											16	**3**	
15×10	23	15	**11**	**18**	**11**	**18**	**11**	**18**	**11**	**18**	**11**	**18**	9.067
	25	10	**12**	**15**	12	16	**12**	**15**	**12**	**15**	**12**	**15**	
			18	**8**	18	11	18	10	18	10	**18**	**8**	
							20	9	20	9	**20**	**6**	

Table 3 Number of non-dominate solution (NS) and quality metric (QM) for 5 Kacem instances

Instance	PF	AL+CGA		MOGA		MOPSO+LS		HSFLA		EPABC		PGDHS	
(n×m)		NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%
4×5	3	2/0	0.0	3/3	100.0	3/3	100.0	3/3	100.0	3/3	100.0	3/3	100
8×8	4	2/0	0.0	3/3	75.0	2/2	50.0	2/2	50.0	2/2	50.0	4/4	100
10×7	3	2/0	0.0	3/2	66.7	3/2	66.7	3/2	66.7	3/2	66.7	3/3	100
10×10	7	4/3	42.9	5/5	71.4	5/5	71.4	5/5	71.4	5/5	71.4	7/7	100
15×10	4	2/0	0.0	3/3	75.0	3/1	25.0	4/2	50.0	4/2	50.0	4/4	100
Sum/Ave	21	12/3	8.6	17/16	77.6	16/13	62.6	17/14	67.6	17/14	67.6	21/21	100

Table 4 Diversification metric (DM) for 5 Kacem instances

Instance	AL+CGA			MOGA			MOPSO+LS			HSFLA			EPABC			PGDHS		
(n×m)	min	ave	sd	min	ave	sd	min	ave	sd	min	ave	sd	min	ave	sd	min	ave	sd
4×5	0.94	1.33	0.38	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
8×8	0.35	0.90	0.43	0.00	0.28	0.56	0.00	1.01	1.25	0.00	1.01	1.25	0.00	1.01	1.25	0.00	0.00	0.00
10×7	0.33	0.87	0.93	0.00	0.25	0.43	0.00	0.25	0.43	0.00	0.25	0.43	0.00	0.25	0.43	0.00	0.00	0.00
10×10	0.00	0.26	0.34	0.00	0.16	0.27	0.00	0.16	0.27	0.00	0.16	0.27	0.00	0.16	0.27	0.00	0.00	0.00
15×10	1.60	2.32	0.72	0.00	0.18	0.35	0.00	0.59	0.59	0.00	0.31	0.38	0.00	0.31	0.38	0.00	0.00	0.00

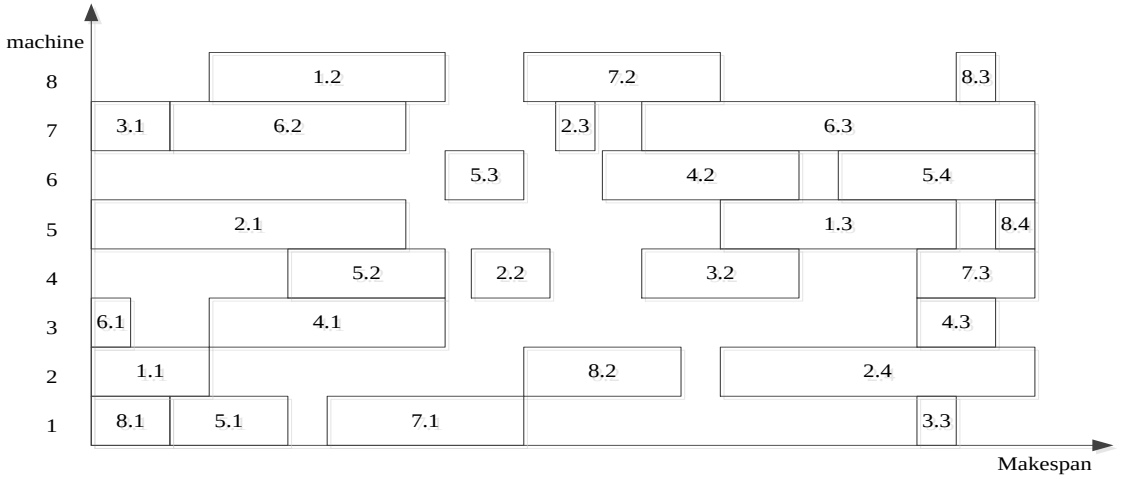

Fig.8 Gantt chart of the solution (24, 3) for Kacem instance 8×8

5.2.2 Test on BRdata instances

In this section, PGDHS algorithm is compared with four very recent algorithms, i.e. Multi-Agent [37], MOGA [36], HSFLA [23] and EPABC [34] using BRdata set. The results obtained by compared algorithms are given in Table 5. All the Pareto solutions in Table 4 are also in boldface.

As shown in Table 5, the proposed PGDHS algorithm is better than or competitive to four compared algorithms**.** 1) For instance MK01, all five algorithms can locate Pareto solution (42, 3). Except Multi-Agent algorithm, other four algorithms can also obtain Pareto solution (40, 4). For instance MK 03, all compared algorithms obtain Pareto solution (204, 21). But, only PGDHS algorithm obtains Pareto solutions (220, 20) and (225, 18). 2) Multi-Agent can locate only one non-dominated solution (42, 3) for instance MK 01 and (204, 21) for instance MK03. 3) MOGA, HSFLA and EPABC obtain two Pareto solutions, (26, 5) and (27, 3) for instance MK 02 and (173, 59) and (174, 58) for instance MK 05. However, PGDHS obtains one more Pareto solution (31, 2) for instance MK 02 and obtains four more Pareto solutions (178, 41), (180, 38), (181, 37) and (190, 36) for instance MK 05**.** 4) MOGA discovers a Pareto solution (60, 19) for instance MK 04 and a Pareto solution (139, 26) for instance MK 07. These two Pareto solutions are not found by PGDHS and other compared algorithms. However, PGDHS obtains four Pareto solutions for instance MK 04 and three of them are not found by MOGA. At the same time, PGDHS discovers five Pareto solutions for instance MK 07 and two of them are not found by MOGA. 5) PGDHS obtains one Pareto solution (62, 2) for instance MK 06. Meanwhile, PGDHS identifies two Pareto solutions (523, 171) and (525, 165) for instance MK 08. But, other compared algorithms could not find any non-dominated solution for these two instances. 6) For instance MK 09 and MK 10, PGDHS obtains five and four Pareto solutions respectively while other compared algorithms could not find any Pareto solutions for these two instances. Table 5 also shows the CPU times for each instance.

Table 6 shows the number of non-dominated solutions and quality metric values of all compared algorithm for 10 BRdata instances. For each compared algorithm, the number of Pareto solutions, the total number of non-dominated solutions, the number of non-dominated solutions which belong to the overall combined PF and the average quality metric values are also calculated. It can be seen from Table 6 that the proposed PGDHS algorithm has 35 non-dominated solutions. This total count is more than the number of solutions obtained by other compared algorithms (Multi-Agent 2, MOGA 13, HSFLA 10 and EPABC 10). The PGDHS algorithm also obtained maximum Pareto solutions compared to other algorithms. For 10instances, PGDHS algorithm found 35 out of 38 Pareto solutions in the overall combined PF. The quality metric values obtained by PGDHS are 100 for seven out of ten instances. The corresponding values for MK04, MK05 and MK07 are 80.00, 85.71 and 83.33, respectively. The average quality metric value of PGDHS is the best one among the compared algorithms.

Table 7 shows the diversification metric values of each compared algorithms for 10 BRdata instances. The minimum, mean and standard deviation of IGD diversification metric values of each algorithm are also calculated. For 10 instances, the minimum values of PGDHS are all zero. At the same time, the mean and standard deviation values of PGDHS are also zero for 7 instances except MK04, MK05 and MK07 instances. For instances MK05 and MK07, the mean and standard deviation values are (0.06, 0.15) and (0.09, 0.22), respectively. These results are better than four compared algorithms even though the values are larger than zero. For instance MK04, PGDHS can also obtain the best mean value, 0.18, while the standard deviation value, 0.40, is worse than four compared algorithms. Four compared algorithms obtain the best minimum value for instance MK01. MOGA, HSFLA and EPABC can also obtain the best mean and standard deviation values for instance MK01. In addition, these three algorithms can also find the best minimum for instance MK02, MK03, MK04 and MK05. However, PGDHS algorithm has the best competitive values for minimum, mean and standard deviation of 10 instances because most results generated by PGDHS are better than those of the four compared algorithms.

Once again, it verifies that the PGDHS possesses superior performance than the four compared algorithms in terms of several widely used performance measures. As an example, Figure 9 shows the Gantt chart of Pareto solution (40, 4) of MK01 instance 10×6 in which the due dates of job1-10 are 30, 28, 31, 25, 42, 33, 19, 36, 33 and 30.

Table 5 Comparison with four existing algorithms based on Makespan and E/T objectives

instance	$n\times m$	Multi-Agent		MOGA		HSFLA		EPABC		PGDHS		CPU (s)
		C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	C_M	E/T	
MK01	10×6			**40**	**4**	**40**	**4**	**40**	**4**	**40**	**4**	5.286

		42	**3**	**42**	**3**	**42**	**3**	**42**	**3**	**42**	**3**	
MK02	10×6	28	3	**26**	**5**	**26**	**5**	**26**	**5**	**26**	**5**	5.394
				27	**3**	**27**	**3**	**27**	**3**	**27**	**3**	
										31	**2**	
MK03	15×8	**204**	**21**	**204**	**21**	**204**	**21**	**204**	**21**	**204**	**21**	24.103
										220	**20**	
										225	**18**	
MK04	15×8	67	17	**60**	**19**	**62**	**15**	**62**	**15**	**62**	**15**	14.734
		70	16	**62**	**15**	68	12	68	12	**65**	**12**	
				68	12	71	11	73	11	**68**	**10**	
										69	**8**	
MK05	15×4	174	61	**173**	**59**	**173**	**59**	**173**	**59**	**173**	**59**	9.010
		178	58	**174**	**58**	**174**	**58**	**174**	**58**	**174**	**58**	
				176	**56**	**176**	**56**	**176**	**56**	**178**	**41**	
				184	49	182	52	179	48	**180**	**38**	
										181	**37**	
										190	**36**	
MK06	10×15	67	7	62	9	64	5	64	5	**62**	**4**	25.775
		77	6	64	5	66	4	66	4			
				66	4							
MK07	20×5	144	23	**139**	**26**	**141**	**21**	**141**	**21**	**140**	**23**	13.914
		146	20	**140**	**23**	**144**	**20**	**144**	**20**	**141**	**21**	
				141	**21**					**144**	**20**	
				144	**20**					**150**	**16**	
										152	**14**	
MK08	20×10	523	177	523	177	523	177	523	177	**523**	**171**	53.626
		525	171	525	171	525	171	525	171	**525**	**165**	
		526	170	526	170	526	170	526	170			
		527	169	527	169	527	169	527	169			
MK09	20×10	311	110	310	112	311	110	309	114	**307**	**96**	62.443
				311	110			310	112	**316**	**80**	
								311	110	**320**	**77**	
										333	**68**	
										336	**65**	
MK10	20×15	215	51	214	53	215	51	215	48	**211**	**48**	78.967
		217	45	216	45	217	45	217	45	**212**	**40**	
		218	41	218	41	218	41	218	41	**213**	**36**	
										215	**31**	

Table 6 Number of non-dominate solutions (NNdS) and quality metric (QM) of compared algorithms

Instance	PF	Multi-Agent		MOGA		HSFLA		EPABC		PGDHS	
		NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%	NS/PF	QM%
MK01	2	1/1	50.00	2/2	100.00	2/2	100.00	2/2	100.00	2/2	100.00
MK02	3	1/0	0.00	2/2	66.7	2/2	66.7	2/2	66.7	3/3	100.00
MK03	3	1/1	33.33	1/1	33.33	1/1	33.33	1/1	33.33	3/3	100.00
MK04	5	2/0	0.00	3/2	40.00	3/1	20.00	3/1	20.00	4/4	80.00

MK05	7	2/0	0.00	4/3	42.86	4/3	42.86	4/3	42.86	6/6	85.71
MK06	1	2/0	0.00	3/0	0.00	2/0	0.00	2/0	0.00	1/1	100.00
MK07	6	2/0	0.00	4/4	66.67	2/2	33.33	2/2	33.33	5/5	83.33
MK08	2	4/0	0.00	4/0	0.00	4/0	0.00	4/0	0.00	2/2	100.00
MK09	5	1/0	0.00	2/0	0.00	1/0	0.00	3/0	0.00	5/5	100.00
MK10	4	3/0	0.00	3/0	0.00	3/0	0.00	3/0	0.00	4/4	100.00
Sum/Ave	38	19/2	8.33	28/14	34.96	24/11	29.62	26/11	29.62	35/35	94.90

Table 7 Diversification metric (DM) for 10 BRdata instances

Instance	Multi-Agent			MOGA			HSFLA			EPABC			PGDHS		
	min	mean	sd	min	mean	sd	min	mean	sd	min	mean	sd	min	mean	sd
MK01	0.00	0.56	0.79	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.00**	**0.00**	**0.00**
MK02	0.33	0.78	0.39	0.00	0.46	0.79	0.00	0.46	0.79	0.00	0.46	0.79	**0.00**	**0.00**	**0.00**
MK03	0.00	4.14	3.69	0.00	4.14	3.69	0.00	4.14	3.69	0.00	4.14	3.69	**0.00**	**0.00**	**0.00**
MK04	1.08	1.30	0.24	0.00	0.36	0.37	0.00	0.52	0.34	0.00	0.54	0.36	**0.00**	**0.18**	0.40
MK05	0.32	1.87	1.43	0.00	0.99	0.94	0.00	1.20	1.15	0.00	0.91	0.94	**0.00**	**0.06**	**0.15**
MK06	5.83	5.83	0.00	2.24	2.24	0.00	2.24	2.24	0.00	2.24	2.24	0.00	**0.00**	**0.00**	**0.00**
MK07	0.33	0.82	0.37	0.00	0.48	0.76	0.00	0.69	0.68	0.00	0.69	0.68	**0.00**	**0.09**	**0.22**
MK08	1.00	1.62	0.87	1.00	1.62	0.87	1.00	1.62	0.87	1.00	1.62	0.87	**0.00**	**0.00**	**0.00**
MK09	2.91	7.12	2.94	2.91	7.12	2.94	2.91	7.12	2.94	2.91	7.12	2.94	**0.00**	**0.00**	**0.00**
MK10	1.25	1.79	0.59	1.46	1.84	0.53	1.25	1.79	0.59	1.00	1.73	0.67	**0.00**	**0.00**	**0.00**

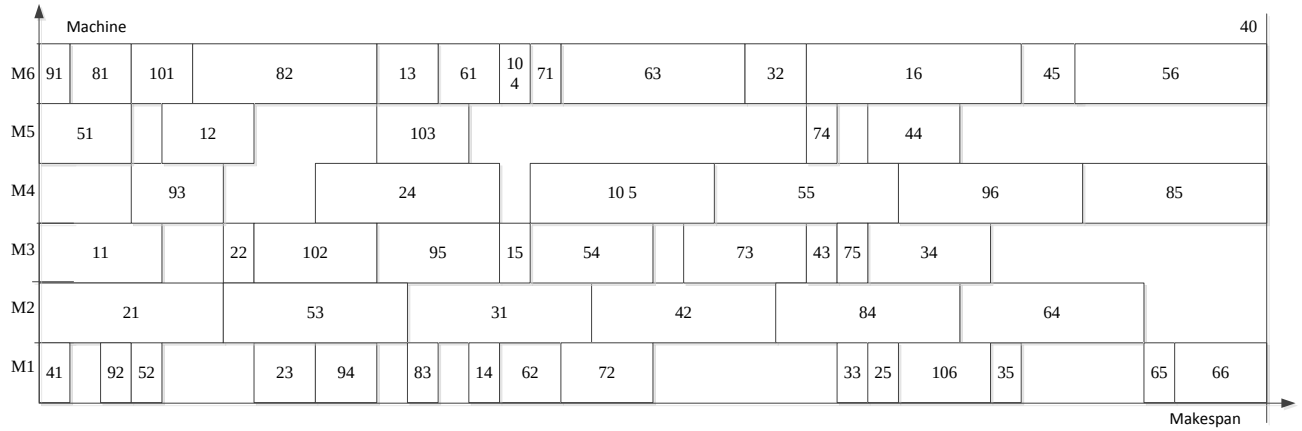

Fig.9 Gantt chart of solution (40, 4) for MK01 instance 10×6

6. Conclusions

This paper proposed a Pareto-based grouping discrete harmony search algorithm (PGDHS) for multi-objective flexible job shop scheduling with makespan, and earliness-tardiness criteria. Two heuristics were developed for initializing the harmony memory. Sub-Harmony memory was employed to improve the diversity of harmony for large population in harmony memory. A method to improvise a new harmony was developed based on the problem characteristics and solution representation. In order to improve the search ability and reduce the search space, two local search algorithms were proposed based on critical path and due date for makespan and E/T criterion, respectively. The large population and local search algorithms balance the capability of exploration and exploitation in PGDHS. It has been

demonstrated by simulation results based on the widely studied benchmarks and comparisons with the state-of-the-art algorithms that the proposed PGDHS is better than the existing algorithms in terms of quality and the number of obtained overall non-dominated solutions. Our future work is to improve the convergence capability of PGDHS by considering additional local search methods and to apply the PGDHS to solve dynamic flexible job shop scheduling problems and other classes of scheduling problems.

Acknowledgement

This research is supported by Remanufacturing Scheduling Systems of A*Star (Agency for Science, Technology and Research, Singapore) under Grant 112 290 4021, National Science Foundation of China under Grant 61174187, Basic Scientific Research Foundation of Northeast University under Grant N110208001 and the Starting Foundation of Northeast University under Grant 29321006.

References

[1] P. Brandimarte, Routing and scheduling in a flexible job shop by tabu search, Annals of Operations Research 41(1993) 157-183.
[2] P. Brucker, R. Schlie, Job-shop scheduling with multi-purpose machines, Computing 45(1990) 369-375.
[3] S. Das, A. Mukhopadhyay, A. Roy, A. Abraham, B.K.R. Panigrahi, Exploratory power of the harmony search algorithm: analysis and improvements for global numerical optimization, IEEE Transactions on System, Man, and Cybernetics, Part B: Cybernetics 41(2011) 89-106.
[4] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6(2002) 182-197.
[5] P. Fattahi, A hybrid multi objective algorithm for flexible job shop scheduling, International Journal of Mathematical and Statistical Sciences 1(2009) 21-26.
[6] J. Gao, L. Sun, M. Gen, A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems, Computers and Operations Research 35(2008) 2892–2907.
[7] K.Z. Gao, Q.K. Pan, J.Q. Li, Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion, International Journal of Advanced Manufacturing Technology 56(2011) 683-692.
[8] K.Z. Gao, Q.K. Pan, J.Q. Li, J.J. Liang, A hybrid harmony search algorithm for the no-wait flow shop scheduling problems, Asia-Pacific J of Operational research 29(2012) 1250012 1-23.
[9] K.Z. Gao, P.N. Suganthan, T.J. Chua, Pareto-based discrete harmony search algorithm for flexible job shop scheduling, Proc. 12th Int. Conf. intelligent system design and applications, ISDA'12, Kochi, India, 2012, pp. 953-956.
[10] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flow hop and job shop scheduling, Mathematics of Operations Research 1(1976) 117–129.
[11] Z.W. Geem, J.H. Kim, G.V. Loganathan, A new heuristic optimization algorithm: harmony search, Simulation 76(2001) 60–68.
[12] Z.W. Geem, Optimal scheduling of multiple dam system using harmony search algorithm, Lecture Notes in Computer Science 4507(2007) 316–323.
[13] M. Gholami, M. Zandieh, Integrating simulation and genetic algorithm to scheduling a dynamic flexible job shop, Journal of Intelligent Manufacturing 20(2009) 481-498.
[14] V.L. Huang, S.Z. Zhao, R, Mallipeddi and P.N. Suganthan, Multi-objective optimization using self-adaptive differential evolution algorithm, Proc. of 2009 IEEE congress on evolutionary computation, IEEE CEC'09, Trondheim, Norway, 2009, pp. 190-194.
[15] A.S. Jain, S. Meeran, Deterministic job-shop scheduling: past, present and future, European Journal of Operational Research 113 (1998) 390–434.
[16] I. Kacem, S. Hammadi, P. Borne, Approach by localization and multi-objective evolutionary optimization for flexible job shop scheduling problems, IEEE transaction on system, Man, and Cybernetics, 32(2002) 1-13.
[17] I. Kacem, S. Hammadi, P. Borne, Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic, Mathematics and Computers in Simulation 60 (2002) 245–276.
[18] K.S. Lee, Z.W. Geem, S.H. Lee, K.W. Bae, The harmony search heuristic algorithm for discrete structural optimization, Engineering Optimization 37 (2005) 663–684.
[19] H. Li, Q. Zhang, Multiobjective optimization problem with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Transactions on Evolutionary Computation 13(2009) 284-302.
[20] J.Q. Li, Q.K. Pan, Y.C. Liang. An effective hybrid tabu search algorithm for multi-objective flexible job shop scheduling problems, Computers and Industrial Engineering 59(2010) 647-662.
[21] J.Q. Li, Q.K. Pan, K.Z. Gao. Pareto-based discrete artificial bee colony algorithm for multi-objective flexible job shop scheduling problems, International Journal of Advanced Manufacturing Technology 55(2011) 1159-1169.
[22] J.Q. Li, Q.K. Pan, S.X. Xie, An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling

problems, Applied Mathematics and Computation 218(2012) 9353-9375.
[23] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, Proc. 2005 IEEE Conf. swarm intelligence symposium, SIS'05, Pasadena, California, USA, 2005, pp. 124-129.
[24] M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, Applied Mathematics and Computation 188(2007) 1567–1579.
[25] G. Moslehi, M. Mahnam, A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search, International Journal of Production Economics 129 (2011) 14-22.
[26] M.G.H. Omran, M. Mahdavi, Global-best harmony search, Applied Mathematics and Computation 198(2008) 643–656.
[27] Q.K. Pan, P.N. Suganthan, J.J. Liang, M.F. Tasgetiren, A local-best harmony search algorithm with dynamic sub-harmony memories for lot-streaming flow shop scheduling problem, Expert System with Applications 38(2011) 3252-3259.
[28] F. Pezzella, G. Morganti, G. Ciaschetti, A genetic algorithm for the flexible job-shop scheduling problem, Computers and Operations Research 35 (2008) 3202–3212.
[29] M. Pinedo, Scheduling: theory, algorithms, and systems, Prentice-Hall, Englewood Cliff, New Jersey, 2002.
[30] M. Rabiee, M. Zandieh, P. Ramezani, Bi-objective partial flexible job shop scheduling problems: NSGA-II, NRGA, MOGA and PAES approaches, International Journal of Production Research 50 (2012)7327-7342.
[31] Seyed Habib A. Rahmati, Proposing a Pareto-based Multi-Objective Evolutionary Algorithm to Flexible Job Shop Scheduling Problem, World Academy of Science, Engineering and Technology, 61(2012), 1160-1165.
[32] C.R. Scrich, V.A. Armentano, M. Laguna, Tardiness minimization in a flexible job shop: A tabu search approach, Journal of Intelligent Manufacturing 15(2004) 103-115.
[33] G. Vilcot, J.C. Billaut, A tabu search algorithm for solving a multi-criteria flexible job shop scheduling problem, International Journal of Production Research 49(2011) 6963-6980.
[34] L. Wang, G. Zhou, Y. Xu, M. Liu. An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job shop scheduling, International Journal of Advanced Manufacturing Technology 60(2012) 1111-1123.
[35] L. Wang, Q.K. Pan, M.F. Tasgetiren, Minimizing the total flow time in a flow shop with blocking by using hybrid harmony search algorithms, Expert System with Applications 37(2010) 7929-7936.
[36] Ling Wang , Shengyao Wang, Min Liu, A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem, International Journal of Production Research 51(2013) 3574-3592
[37] X. Wang, L. Gao, G. Zhang, X. Shao, A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem, International Journal of Advanced Manufacturing Technology 51(2010) 757–767.
[38] Z.B. Wu, M.X. Weng, Multi-agent scheduling method with earliness and tardiness objectives in flexible job shops, IEEE Transactions on System, Man, and Cybernetics-Part B: Cybernetics 35(2005), 293-301.
[39] W.J. Xia, Z.M. Wu, An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems, Computers and Industrial Engineering 48(2005) 409–425.
[40] L.N. Xing, Y.W. Chen, K.W. Yang, Multi-objective flexible job shop schedule: Design and evaluation by simulation modeling, Applied Soft Computing 9 (2009) 362–376.
[41] L.N. Xing, Y.W. Chen, K.W. Yang, An efficient search method for multi-objective flexible job shop scheduling problems, Journal of Intelligent Manufacturing 20(2009) 283–293.
[42] Ali R. Yildiz, A comparative study of population-based optimization algorithms for turning operations, Information Sciences 210 (2012), 81-88.
[43] Ali R. Yildiz, Hybrid Taguchi-differential evolution algorithm for optimization of multi-pass turning operations, Applied Soft Computing 13(2013) 1433-1439.
[44] Ali R. Yildiz, A new hybrid differential evolution algorithm for the selection of optimal machining parameters in milling operations, Applied Soft Computing 13(2013) 1561-1566.
[45] Ali R. Yildiz, Cuckoo search algorithm for the selection of optimal machining parameters in milling operations, International Journal of Advanced Manufacturing Technology 64(2013) 55-61.
[46] G.H. Zhang, X.Y. Shao, P.G.Li, L. Gao, An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, Computers and Industrial Engineering 56(2009) 1309–1318.
[47] G.H. Zhang, L. Gao, Y. Shi, An effective genetic algorithm for the flexible job-shop scheduling problem, Expert System with Application 38(2011) 3563-3573.
[48] C.Y. Zhang, P.G. Li, Z.L. Guan, Y.Q. Rao, A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem, Computers and Operations Research 34 (2007) 3229–3242.
[49] S. Zhao, P. N. Suganthan, Q. Zhang, Decomposition-based multi-objective evolutionary algorithm with an ensemble of neighborhood sizes, IEEE Transactions on Evolutionary Computation 16(2012) 442-446.